

# PyStruct

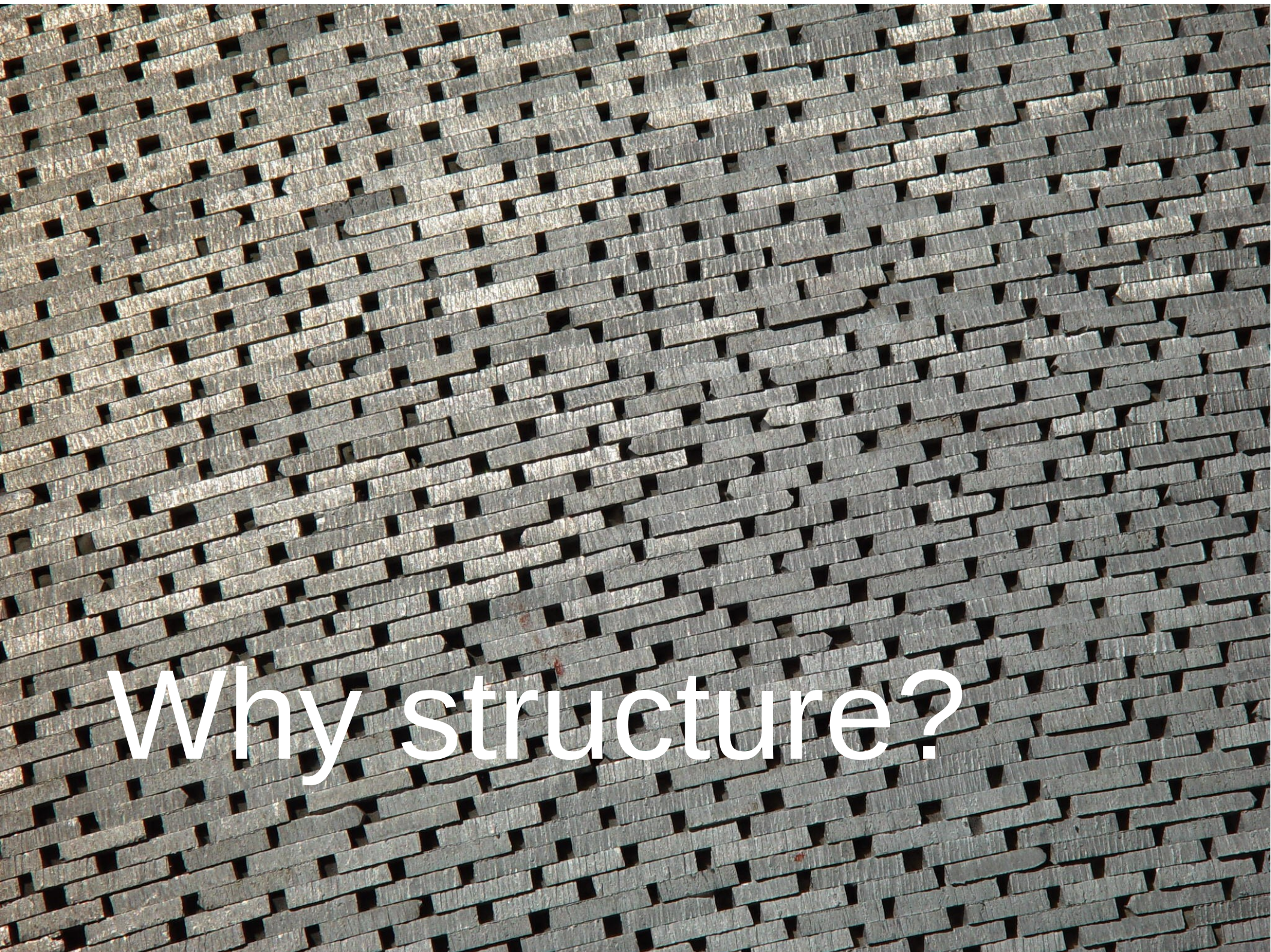
## Structured Prediction in Python

Andreas Mueller (NYU Center for Data Science, scikit-learn)

# Structured Prediction

$$y = (y_1, y_2, \dots, y_{n_k})$$





Why structure?



# Applications: Multi-Label Classification

---

	Politics	Sports	Finance	Domestic	Religion
News Story1	1	0	0	1	1
News Story2	0	1	0	1	0
News Story3	0	0	1	0	0

---

# Applications: Multi-Label Classification

---

	Politics	Sports	Finance	Domestic	Religion
News Story1	1	0	0	1	1
News Story2	0	1	0	1	0
News Story3	0	0	1	0	0

---

---

	Owns Car	Smokes	Married	Self-Employed	Has Kids
Customer1	1	0	1	0	1
Customer2	1	1	0	1	0
Customer3	0	1	1	0	0

---

# Applications: Sequence Tagging



# Applications: Sequence Tagging



Stroke cat.



Stroke cat.



Stroke cat.

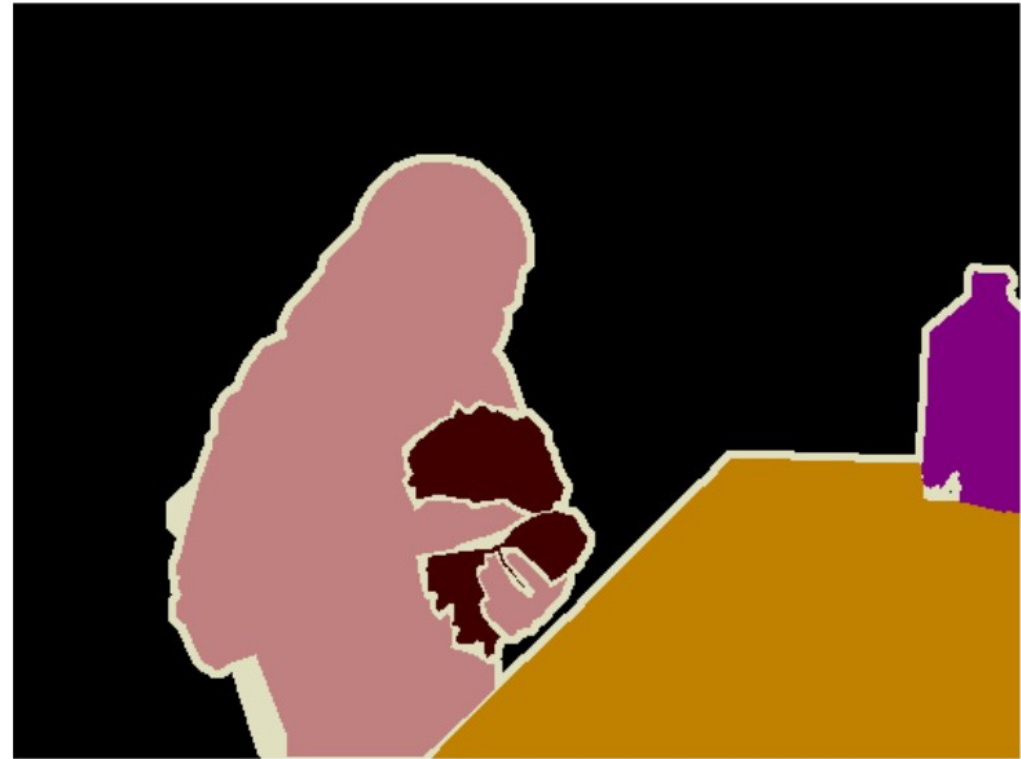


Open trash can.



Put cat in trash can.

# Applications: Image Segmentation







# The Essence of Structured Prediction

$$f(x, w) := \arg \max_{y \in \mathcal{Y}} g(x, y, w)$$

# The Essence of Structured Prediction

$$f(x, w) := \arg \max_{y \in \mathcal{Y}} g(x, y, w)$$

If you like:

$$\arg \max_{y \in \mathcal{Y}} p(y|x, w)$$



# Pairwise Structured Models

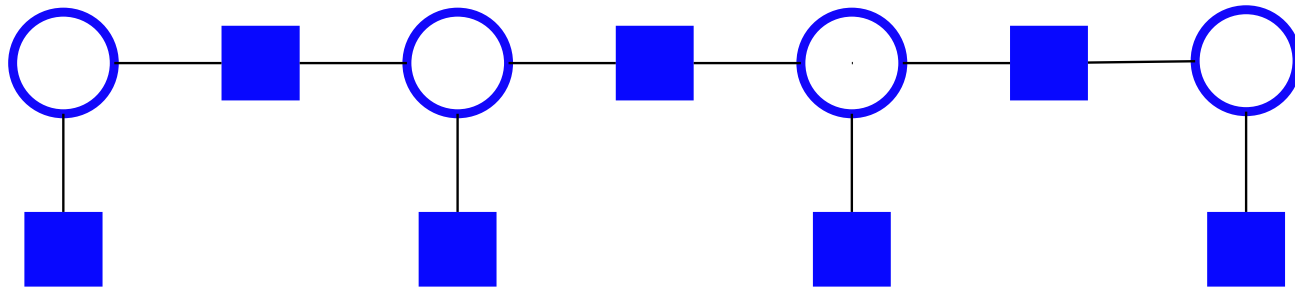
$$\arg \max_{y_1, y_2, \dots, y_n} w^T \psi(x, y)$$

$$= \arg \max_{y_1, y_2, \dots, y_n} \sum_I w_i^T \psi(x, y_i) + \sum_{(i,j) \in E} w_{i,j}^T \psi(x, y_i, y_j)$$

# Pairwise Structured Models

$$\arg \max_{y_1, y_2, \dots, y_n} w^T \psi(x, y)$$

$$= \arg \max_{y_1, y_2, \dots, y_n} \sum_I w_i^T \psi(x, y_i) + \sum_{(i,j) \in E} w_{i,j}^T \psi(x, y_i, y_j)$$



# PyStruct Architecture

Estimator = Learner + Model + Inference

$$\arg \max_{y_1, y_2, \dots, y_n} w^T \psi(x, y)$$



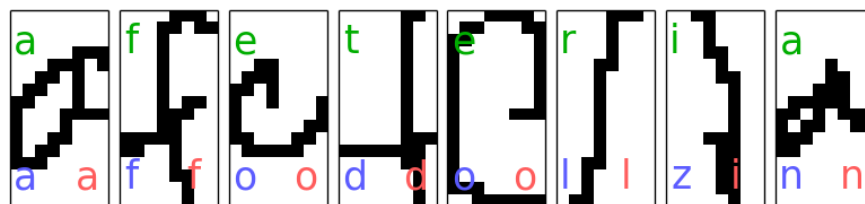
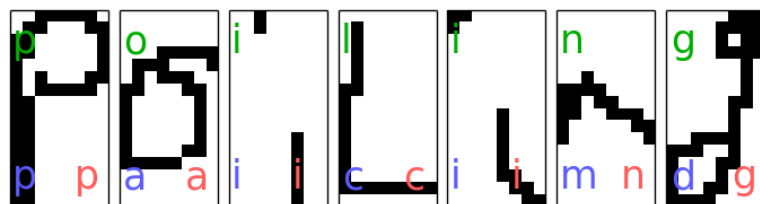
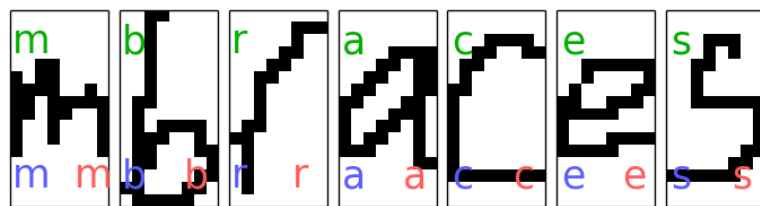
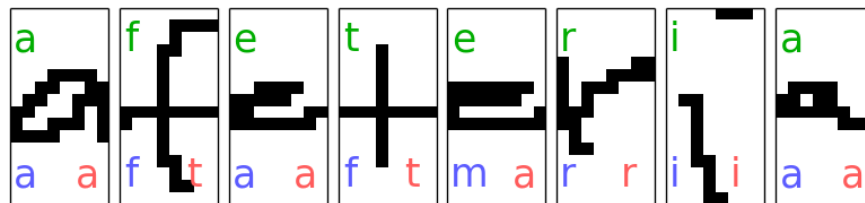
# PyStruct Architecture

Estimator = Learner + Model + Inference

$$\arg \max_{y_1, y_2, \dots, y_n} w^T \psi(x, y)$$

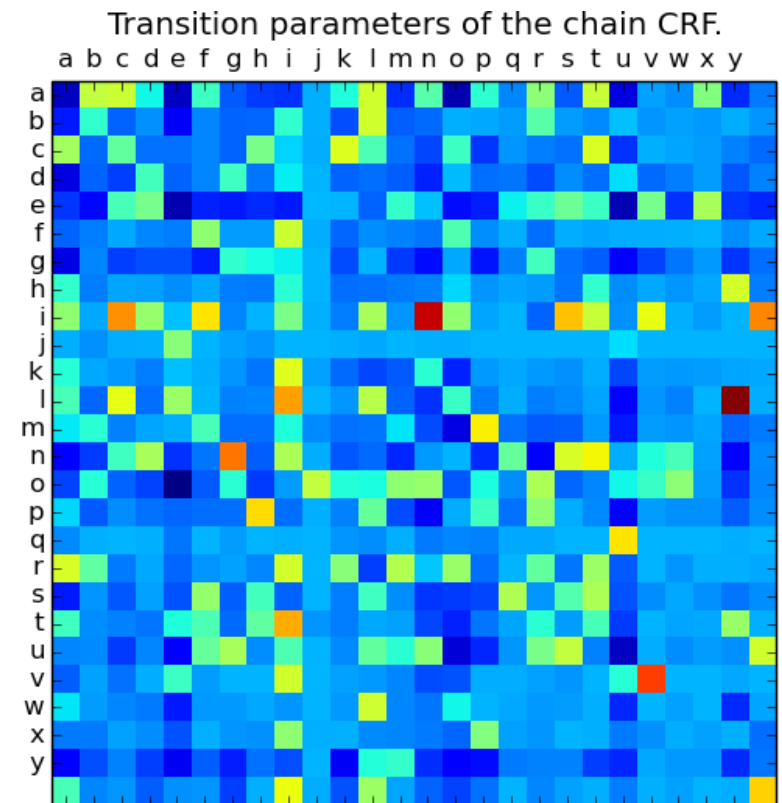
```
model = ChainCRF(inference="max_product")
ssvm = OneSlackSSVM(model=model, C=.1, inference_cache=50,,
                    tol=0.1, verbose=3)
ssvm.fit(X_train, y_train)
```

# Sequence Tagging example



```
model = ChainCRF(inference="max_product")
ssvm = OneSlackSSVM(model=model, C=.1, inference_cache=50,,
                    tol=0.1, verbose=3)
ssvm.fit(X_train, y_train)
```

# Sequence Tagging example

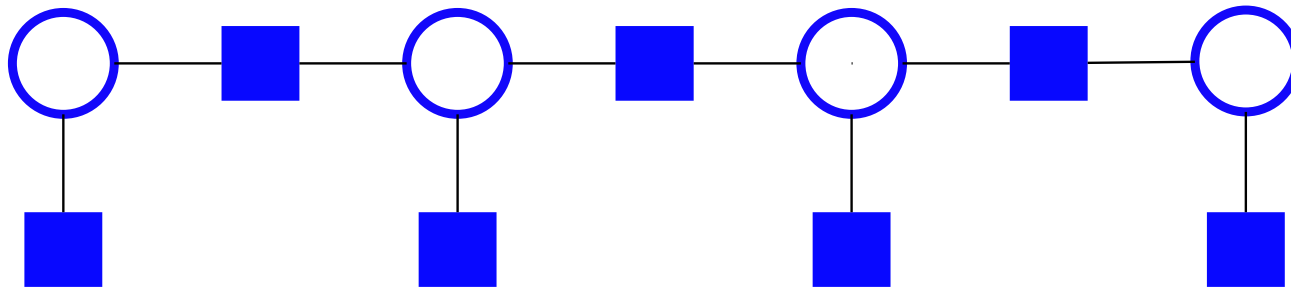


```
model = ChainCRF(inference="max_product")
ssvm = OneSlackSSVM(model=model, C=.1, inference_cache=50,,
                    tol=0.1, verbose=3)
ssvm.fit(X_train, y_train)
```



# The Devil is in the Inference

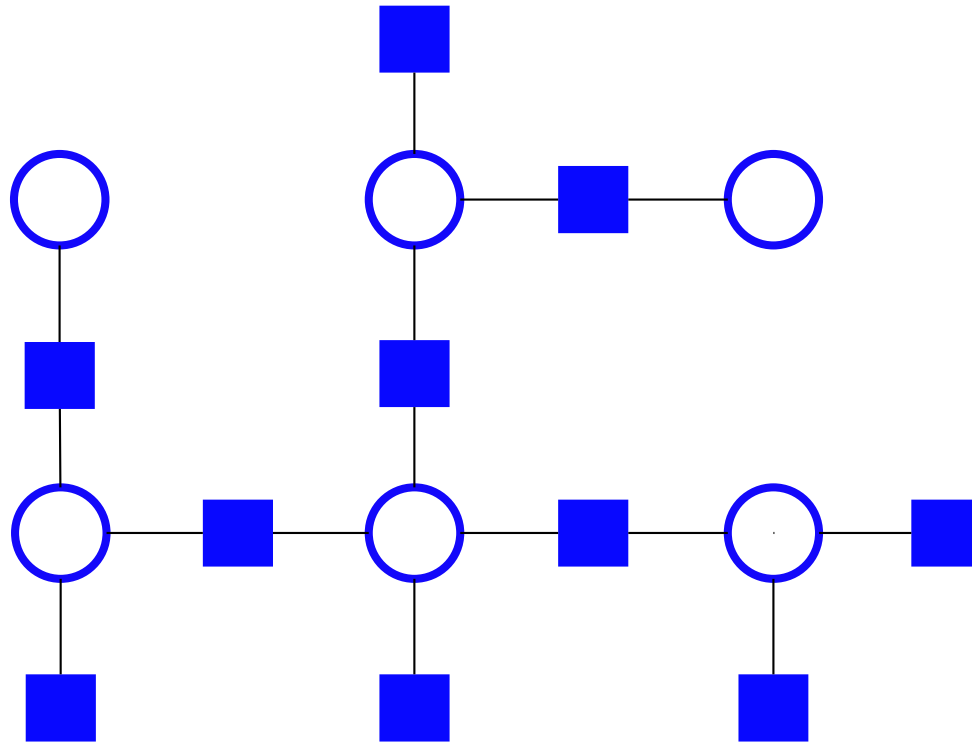
$$\arg \max_{y_1, y_2, \dots, y_n} w^T \psi(x, y)$$



Easy: Dynamic Programming

# The Devil is in the Inference

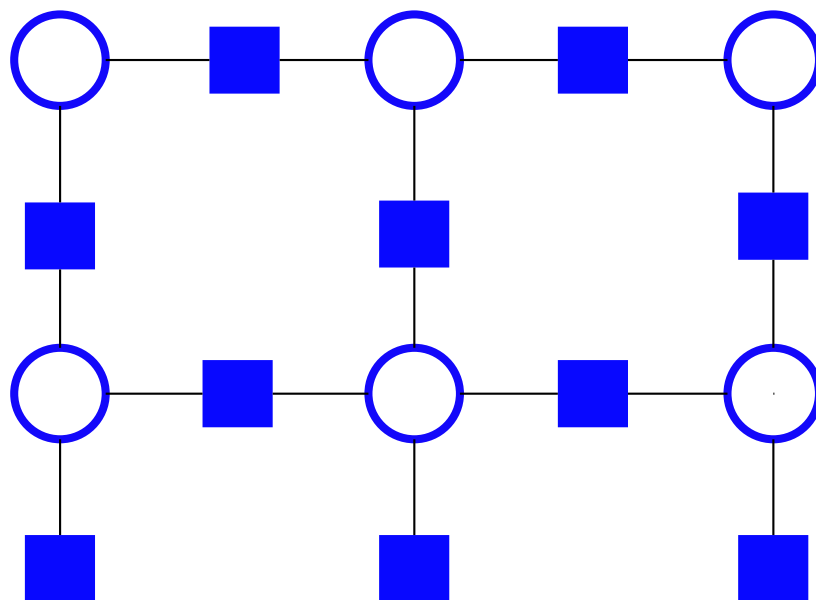
$$\arg \max_{y_1, y_2, \dots, y_n} w^T \psi(x, y)$$



Easy: Dynamic Programming

# The Devil is in the Inference

$$\arg \max_{y_1, y_2, \dots, y_n} w^T \psi(x, y)$$

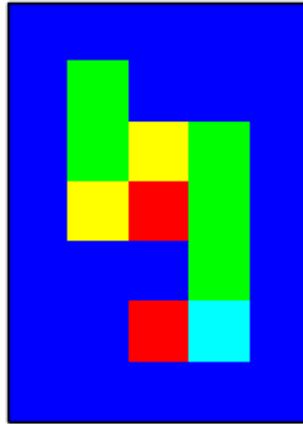


HARD!

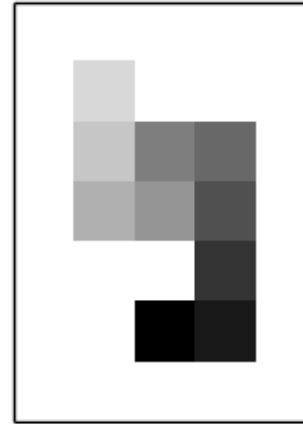
AD3, QPBO, LP, Loopy BP, ....

# Grid Graphs: Snakes

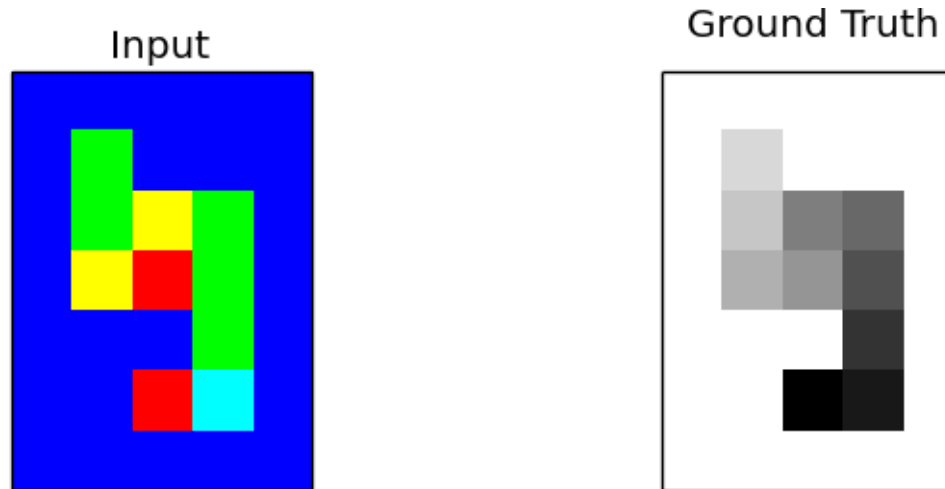
Input



Ground Truth



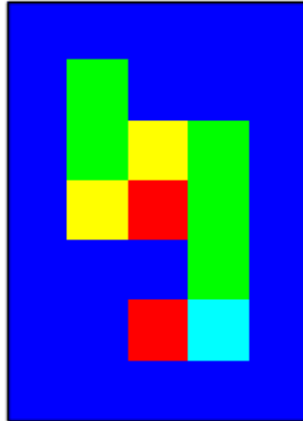
# Grid Graphs: Snakes



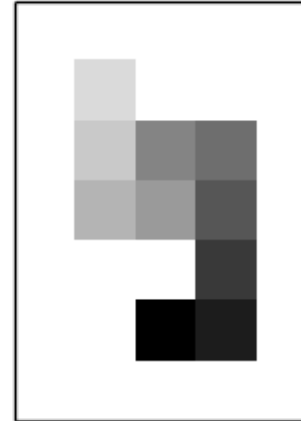
```
crf = EdgeFeatureGraphCRF(inference_method='qpbo')
ssvm = OneSlackSSVM(crf, inference_cache=50, C=.1, tol=.1,
                    switch_to='ad3', n_jobs=1)
ssvm.fit(X_train_edge_features, Y_train_flat)
```

# Grid Graphs: Snakes

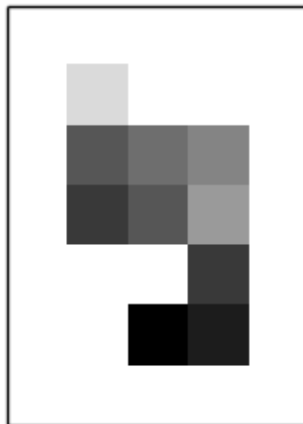
Input



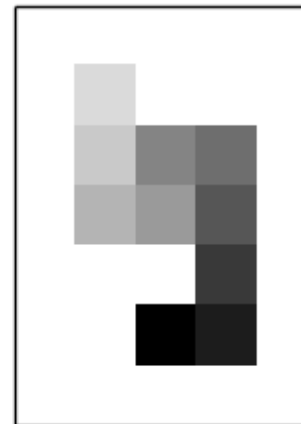
Ground Truth



Prediction w/o edge features



Prediction with edge features



# Implemented Methods

Estimator = Learner + Model + Inference

- Learner: SubgradientSSVM, StructuredPerceptron, OneSlackSSVM, LatentSSVM
- Model: BinaryClf, MultiLabelClf, ChainCRF, GraphCRF, EdgeFeatureGraphCRF
- Inference: Linear Programming, QPBO (PyQPBO), Dual Decomposition (AD3), Message Passing, everything (OpenGM)



# Classes of Inference Algorithms

## **Exact Algorithms**

Max-Product (Chains, Trees) 'max-product'

Exhaustive (usually too expensive)

Relaxed algorithms + branch & bound  
( 'ad3', { 'branch\_and\_bound': True } )

## **Relaxed**

Linear Programming (sloooow) 'lp'

Dual Decomposition 'ad3'

## **Approximate / heuristics**

Loopy message passing 'max-product'

QPBO 'qpbo'

# Classes of Inference Algorithms

## **Exact Algorithms**

Max-Product (Chains, Trees) 'max-product'

Exhaustive (usually too expensive)

Relaxed algorithms + branch & bound  
( 'ad3', { 'branch\_and\_bound': True } )

## **Relaxed**

Linear Programming (sloooow) 'lp'

Dual Decomposition 'ad3'

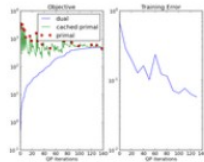
## **Approximate / heuristics**

Loopy message passing 'max-product'

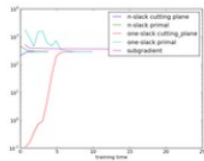
QPBO 'qpbo'

Install OpenGM for many more!

# Examples



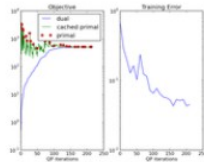
Plotting the objective and constraint caching in 1-slack SSVM



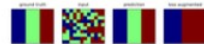
SVM objective values



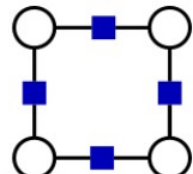
Latent SVM for odd vs. even digit classification



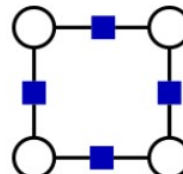
Efficient exact learning of 1-slack SSVMs



Learning directed interactions on a 2d grid



Mult-label classification



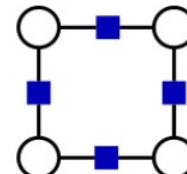
SVM as CRF



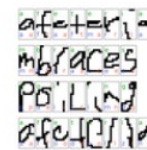
Learning interactions on a 2d grid



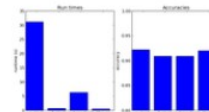
Latent Variable Hierarchical CRF



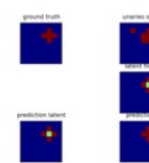
Semantic Image Segmentation on Pascal VOC



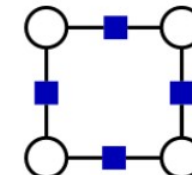
OCR Letter sequence recognition



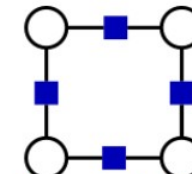
Binary SVM as SSVM



Latent Dynamics CRF



Crammer-Singer Multi-Class SVM



Comparing PyStruct and SVM-Struct

Thank you for your attention.



@t3kcit



@amueller



t3kcit@gmail.com



<http://amueller.github.io>